Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

"theorem proving" "code generation"

US Patent & Trademark Office

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **theorem proving code generation** Found **4,084** of **147,060**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Optimal Code Generation for Expression Trees

A. V. Aho, S. C. Johnson

July 1976 **Journal of the ACM (JACM),** Volume 23 Issue 3

Full text available: pdf(895.62 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper discusses algorithms which transform expression trees into code for register machines. A necessary and sufficient condition for optimality of such an algorithm is derived, which applies to a broad class of machines. A dynamic programming algorithm is then presented which produces optimal code for any machine in this class; this algorithm runs in time linearly proportional to the size of the input.

2 Automatic reasoning about numerical stability of rational expressions

B. W. Char

July 1989 **Proceedings of the ACM-SIGSAM 1989 international symposium on Symbolic and algebraic computation**

Full text available: pdf(686.11 KB) Additional Information: full citation, abstract, references, index terms, review

While numerical (e.g. Fortran) code generation from computer algebra is nowadays relatively easy to do, the expressions as they are produced via computer algebra typically benefit from non-trivial reformulation for efficiency and numerical stability. To assist in automatic "expert reformulation", we desire good automated tools to assess the stability of a particular form of an expression. In this paper, we discuss an approach to proofs of numerical stability (with respect to rou ...

3 Hybrid system for multi-language and multi-environment generation of numerical codes

Joze Korelc

~~July 2001~~ **Proceedings of the 2001 international symposium on Symbolic and algebraic computation**

Full text available: pdf(673.56 KB) Additional Information: full citation, abstract, references, citings, index terms

The paper presents a hybrid system for automatic generation of numerical procedures for various finite element environments from the same symbolic description. The system consists of two major components. The *Mathematica* package *AceGen* is used for the automatic derivation of formulae needed in numerical procedures. An approach, implemented in *AceGen*, avoids the usual problem of uncontrollable growth of expressions by combining several techniques: symbolic and algebraic ca ...

Keywords: code generation, finite element environments, symbolic approach

4 Producing more reliable software: mature software engineering process vs. state-of-the-art technology?
James C. Widmaier
June 2000 **Pr ceedings of the 22nd internati nal conference n Software engineering**
Full text available: pdf(96.31 KB) Additional Information: full citation, abstract, references, citings, index terms

A customer of high assurance software recently sponsored a software engineering experiment in which a real-time software system was developed concurrently by two popular popular software development methodologies. One company specialized in the state-of-the-practice waterfall method rated at a Capability Maturity Model Level 4. A second developer employed his mathematically based formal method with automatic code generation. As specified in separate contracts, C++ code plus development documentatio ...

Keywords: capability maturity model, formal methods, software engineering experiment, software process and product metrics, software reliability

5 Using KIDS as a tool support for VDM
Y. Ledru
May 1996 **Proceedings of the 18th international conference on Software engineering**
Full text available: pdf(1.13 MB) Publisher Site Additional Information: full citation, abstract, references, index terms

KIDS/VDM is an experimental environment that supports the synthesis of executable prototypes from VDM specifications. The development proceeds as a series of correctness preserving transformations under the strict control of the tool. A by-product of this development is the proof of consistency properties of the original specification. Experiments with the tool have shown its ability to handle independently written specifications. It also revealed useful to detect errors in specifications. The e ...

Keywords: KIDS, REFINE, REGROUP, VDM, VDM specifications, correctness preserving transformations, executable prototypes synthesis, formal specification, program verification, proof of consistency, specification languages, theorem prover, theorem proving, tool support

6 Verification of microprogrammed computer architectures in the S*-system: a case study
W. Damm, G. Dohmen
December 1985 **ACM SIGMICRO Newsletter , Proceedings of the 18th annual workshop on Microprogramming**, Volume 16 Issue 4
Full text available: pdf(1.43 MB) Additional Information: full citation, abstract, references, citings, index terms

We apply the verification methodology underlying the S*-System[12], [13] to the verification of a hierarchically structured design [16] of an emulation of the instruction-set of a commercially available computer on a commercially available micro-architecture. Based on this case-study, we discuss some aspects of the relation between verification and generation of microcode.

7 Parsing and compiling using Prolog
Jacques Cohen, Timothy J. Hickey
March 1987 **ACM Transacti ns n Programming Languages and Systems (TOPLAS)**, Volume 9 Issue 2
Full text available: pdf(2.83 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper presents the material needed for exposing the reader to the advantages of using

Prolog as a language for describing succinctly most of the algorithms needed in prototyping and implementing compilers or producing tools that facilitate this task. The available published material on the subject describes one particular approach in implementing compilers using Prolog. It consists of coupling actions to recursive descent parsers to produce syntax-trees which are subsequently utilized ...

8 Session 7: requirements analysis: Secure systems development based on the common criteria: the PalME project
Monika Vetterling, Guido Wimmel, Alexander Wisspeintner
~~November 2002~~ **ACM SIGSOFT Software Engineering Notes**, Volume 27 Issue 6
Full text available: pdf(1.15 MB) Additional Information: full citation, abstract, references, index terms

Security is a very important issue in information processing, especially in open network environments like the Internet. The *Common Criteria* (CC) is the standard requirements catalogue for the evaluation of security critical systems. Using the CC, a large number of security requirements on the system itself and on the system development can be defined. However, the CC does not give methodological support.In this paper, we show how integrate security aspects into the software engineering p ...

Keywords: AutoFocus, CASE, case study, common criteria, development process, formal methods, graphical description techniques, requirements engineering, security engineering, software design, software engineering

9 Requirements analysis: Secure systems development based on the common criteria: the PalME project
Monika Vetterling, Guido Wimmel, Alexander Wisspeintner
~~November 2002~~ **Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering**
Full text available: pdf(640.02 KB) Additional Information: full citation, abstract, references, index terms

Security is a very important issue in information processing, especially in open network environments like the Internet. The Common Criteria (CC)is the standard requirements catalogue for the evaluation of security critical systems. Using the CC, a large number of security requirements on the system itself and on the system development can be defined. However, the CC does not give methodological support. In this paper, we show how integrate security aspects into the software engineering process. ...

Keywords: AutoFocus, CASE, case study, common criteria, development process, formal methods, graphical description techniques, requirements engineering, security engineering, software design, software engineering

10 Code Generation and Storage Allocation for Machines with Span-Dependent Instructions
Edward L. Robertson
January 1979 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 1 Issue 1
Full text available: pdf(652.48 KB) Additional Information: full citation, abstract, references, citings, index terms

Many machine languages have two instruction formats, one of which allows addressing of "nearby" operands with "short" (e.g. one word) instructions, while "faraway" operands require "long" format (e.g. two words). Because the distance between an instruction and its operand depends upon the formats of the intervening instructions, the formats of different instructions may be interdependent. An efficient technique is discussed which op ...

11 Specification, validation, and synthesis of email agent controllers: A case study in function rich reactive system design
Robert J. Hall

August 2000 **Pr ceedings f the third w rkshop on F rmal meth ds in software practice**

Full text available: pdf(527.90 KB) Additional Information: full citation, abstract, references, index terms

With a few exceptions, previous formal methods for reactive system design have focused on finite state machines represented in terms of boolean states and boolean next-state functions. By contrast, in many reactive system domains requirements engineers and developers think in terms of complex data types and expressive next-state functions. Formal methods for reactive system design must be extended to meet their needs as well. I term a reactive system function rich if expr ...

Keywords: Electronic Mail, Formal Methods, Reactive Systems

12 Evolutionary design of complex software (EDCS) demonstration days 1999
Wayne Stidolph
January 2000 **ACM SIGSOFT Software Engineering Notes**, Volume 25 Issue 1

Full text available: pdf(1.90 MB) Additional Information: full citation, abstract, index terms

This report summarizes the Product/Technology demonstrations given at Defense Advanced Research Projects Agency (DARPA) Evolutionary Design of Complex Software (EDCS) Program Demonstration Days, held 28-29 June 1999 at the Sheraton National Hotel, Arlington, VA.

13 Descriptions of prototypes: Interactive and visual environment supporting conceptual modeling of complex OODB applications
M. Missikoff, R. Pizzicannella
May 1996 **Proceedings of the workshop on Advanced visual interfaces**

Full text available: pdf(395.75 KB) Additional Information: full citation, abstract, references, citings

In this paper we present the graphical user interface of Mosaico, an environment for the analysis and conceptual modeling of Object-Oriented database applications. Mosaico is based on a formalism, the Object-Oriented conceptual language TQL++, that appears more friendly than others. Neverthless, to relieve the designer from knowing the details of TQL++, we developed an iconic interface that guides the construction of a database application specification. The output of the conceptual modeling pha ...

14 Specware
Jim McDonald
January 2000 **ACM SIGSOFT Software Engineering Notes**, Volume 25 Issue 1

Full text available: pdf(273.70 KB) Additional Information: full citation, index terms

15 SoBelt: structural and behavioral execution instrumentation tool
Debra Richardson
January 2000 **ACM SIGSOFT Software Engineering Notes**, Volume 25 Issue 1

Full text available: pdf(273.70 KB) Additional Information: full citation, index terms

16 Siddhartha - automated test driver-oracle synthesis
Debra Richardson
January 2000 **ACM SIGSOFT S ftware Engineering N tes**, Volume 25 Issue 1

Full text available: pdf(275.93 KB) Additional Information: full citation, index terms

17 Securely wrapping COTS products

Bob Balzer
January 2000 **ACM SIGSOFT S ftware Engineering N tes**, Volume 25 Issue 1
Full text available: pdf(275.93 KB) Additional Information: full citation, index terms

18 Software architecture, analysis, generation, and evolution (SAAGE)
Barry Boehm, Neno Medvidovic
January 2000 **ACM SIGSOFT Software Engineering Notes**, Volume 25 Issue 1
Full text available: pdf(275.93 KB) Additional Information: full citation, index terms

19 High assurance technologies
Michal Young
January 2000 **ACM SIGSOFT Software Engineering Notes**, Volume 25 Issue 1
Full text available: pdf(275.93 KB) Additional Information: full citation, index terms

20 On the Simplification and Equivalence Problems for Straight-Line Programs
Oscar H. Ibarra, Brian S. Leininger
July 1983 **Journal of the ACM (JACM)**, Volume 30 Issue 3
Full text available: pdf(691.46 KB) Additional Information: full citation, references, index terms

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login
Search: ● The ACM Digital Library ○ The Guide
"theorem proving" code generation

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **theorem proving code generation** Found **48,719** of **147,060**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Hybrid system for multi-language and multi-environment generation of numerical codes
Joze Korelc
~~July 2001~~ **Proceedings of the 2001 international symposium on Symbolic and algebraic computation**
Full text available: pdf(673.56 KB) Additional Information: full citation, abstract, references, citings, index terms

The paper presents a hybrid system for automatic generation of numerical procedures for various finite element environments from the same symbolic description. The system consists of two major components. The *Mathematica* package *AceGen* is used for the automatic derivation of formulae needed in numerical procedures. An approach, implemented in *AceGen*, avoids the usual problem of uncontrollable growth of expressions by combining several techniques: symbolic and algebraic ca ...

Keywords: code generation, finite element environments, symbolic approach

2 Z-Resolution: Theorem-Proving with Compiled Axioms
John K. Dixon
January 1973 **Journal of the ACM (JACM)**, Volume 20 Issue 1
Full text available: pdf(1.46 MB) Additional Information: full citation, abstract, references, citings, index terms

An improved procedure for resolution theorem proving, called Z-resolution, is described. The basic idea of Z-resolution is to "compile" some of the axioms in a deductive problem. This means to automatically transform the selected axioms into a computer program which carries out the inference rules indicated by the axioms. This is done automatically by another program called the specializer. The advantage of doing this is that the compiled axioms run faster, just as a compiled pr ...

3 Optimal Code Generation for Expression Trees
A. V. Aho, S. C. Johnson
July 1976 **Journal of the ACM (JACM)**, Volume 23 Issue 3
Full text available: pdf(695.62 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper discusses algorithms which transform expression trees into code for register machines. A necessary and sufficient condition for optimality of such an algorithm is derived, which applies to a broad class of machines. A dynamic programming algorithm is then presented which produces optimal code for any machine in this class; this algorithm runs in time linearly proportional to the size of the input.

4 Using KIDS as a tool support for VDM
Y. Ledru
May 1996 **Pr ceedings f the 18th internati nal c nference n S ftware engineering**
Full text available: pdf(1.13 MB) Publisher Site Additional Information: full citation, abstract, references, index terms

KIDS/VDM is an experimental environment that supports the synthesis of executable prototypes from VDM specifications. The development proceeds as a series of correctness preserving transformations under the strict control of the tool. A by-product of this development is the proof of consistency properties of the original specification. Experiments with the tool have shown its ability to handle independently written specifications. It also revealed useful to detect errors in specifications. The e ...

Keywords: KIDS, REFINE, REGROUP, VDM, VDM specifications, correctness preserving transformations, executable prototypes synthesis, formal specification, program verification, proof of consistency, specification languages, theorem prover, theorem proving, tool support

5 Automated proofs of object code for a widely used microprocessor
Robert S. Boyer, Yuan Yu
January 1996 **Journal of the ACM (JACM)**, Volume 43 Issue 1
Full text available: pdf(2.41 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: Ada, Boyer-Moore logic, C, Common Lisp, MC68xxx, Nqthm, automated reasoning, formal methods, machine code, mechanical theorem proving, object code, program proving, program verification

6 Verification of microprogrammed computer architectures in the S*-system: a case study
W. Damm, G. Dohmen
December 1985 **ACM SIGMICRO Newsletter , Proceedings of the 18th annual workshop on Microprogramming**, Volume 16 Issue 4
Full text available: pdf(1.43 MB) Additional Information: full citation, abstract, references, citings, index terms

We apply the verification methodology underlying the S*-System[12], [13] to the verification of a hierarchically structured design [16] of an emulation of the instruction-set of a commercially available computer on a commercially available micro-architecture. Based on this case-study, we discuss some aspects of the relation between verification and generation of microcode.

7 Using specialized procedures and specification-based analysis to reduce the runtime costs of modularity
Mark T. Vandevoorde, John V. Guttag
December 1994 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2nd ACM SIGSOFT symposium on Foundations of software engineering**, Volume 19 Issue 5
Full text available: pdf(606.23 KB) Additional Information: full citation, abstract, references, citings, index terms

Managing tradeoffs between program structure and program efficiency is one of the most difficult problems facing software engineers. Decomposing programs into abstractions simplifies the construction and maintenance of software and results in fewer errors. However, the introduction of these abstractions often introduces significant inefficiencies.This paper describes a strategy for eliminating many of these inefficiencies. It is based upon providing alternative implementations of the same abstra ...

Keyw rds: compilers, formal specifications, program modularity, program optimization, software interfaces

8 Code Generation for a One-Register Machine
John Bruno, Ravi Sethi
July 1976 **Journal of the ACM (JACM)**, Volume 23 Issue 3

Full text available: pdf(553.60 KB) Additional Information: full citation, abstract, references, citings, index terms

The majority of computers that have been built have performed all computations in devices called accumulators, or registers. In this paper, it is shown that the problem of generating minimal-length code for such machines is hard in a precise sense; specifically it is shown that the problem is NP-complete. The result is true even when the programs being translated are arithmetic expressions. Admittedly, the expressions in question can become complicated.

9 Automatic generation of program specifications
Jeremy W. Nimmer, Michael D. Ernst
~~July 2002~~ **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis**, Volume 27 Issue 4

Full text available: pdf(154.41 KB) Additional Information: full citation, abstract, references, citings

Producing specifications by dynamic (runtime) analysis of program executions is potentially unsound, because the analyzed executions may not fully characterize all possible executions of the program. In practice, how accurate are the results of a dynamic analysis? This paper describes the results of an investigation into this question, determining how much specifications generalized from program runs must be changed in order to be verified by a static checker. Surprisingly, small test suites cap ...

10 Automated Generation of Test Programs from Closed Specifications of Classes and Test Cases
~~May 2004~~ **Proceedings of the 26th International Conference on Software Engineering**

Full text available: pdf(315.07 KB) Publisher Site Additional Information: full citation, abstract

Most research on automated specification-based softwaretesting has focused on the automated generation oftest cases. Before a software system can be tested, it must beset up according to the input requirements of the test cases.This setup process is usually performed manually, especiallywhen testing complex data structures and databases.After the system is properly set up, a test execution tool runsthe system according to the test cases and pre-recorded testscripts to obtain the outputs, which a ...

11 Producing more reliable software: mature software engineering process vs. state-of-the-art technology?
James C. Widmaier
June 2000 **Proceedings of the 22nd international conference on Software engineering**

Full text available: pdf(98.31 KB) Additional Information: full citation, abstract, references, citings, index terms

A customer of high assurance software recently sponsored a software engineering experiment in which a real-time software system was developed concurrently by two popular popular software development methodologies. One company specialized in the state-of-the-practice waterfall method rated at a Capability Maturity Model Level 4. A second developer employed his mathematically based formal method with automatic code generation. As specified in separate contracts, C++ code plus development documentatio ...

Keyw rds: capability maturity model, formal methods, software engineering experiment,

software process and product metrics, software reliability

12 The design and implementation of a certifying compiler
George C. Necula, Peter Lee
May 1998 **ACM SIGPLAN N tices , Pr ceedings f the ACM SIGPLAN 1998 conference n Pr gramming language design and implementati n**, Volume 33 Issue 5
Full text available: pdf(1.66 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the design and implementation of a compiler that translates programs written in a type-safe subset of the C programming language into highly optimized DEC Alpha assembly language programs, and a *certifier* that automatically checks the type safety and memory safety of any assembly language program produced by the compiler. The result of the certifier is either a formal proof of type safety or a counterexample pointing to a potential violation of the type system by the t ...

13 Optimal code generation for expression trees
A. V. Aho, S. C. Johnson
May 1975 **Proceedings of seventh annual ACM symposium on Theory of computing**
Full text available: pdf(819.72 KB) Additional Information: full citation, abstract, references, citings, index terms

We discuss the problem of generating code for a wide class of machines, restricting ourselves to the computation of expression trees. After defining a broad class of machines and discussing the properties of optimal programs on these machines, we derive a necessary and sufficient condition which can be used to prove the optimality of any code generation algorithm for expression trees on this class. We then present a dynamic programming algorithm which produces optimal code for any machine i ...

14 Mobile code: The source is the proof
Vivek Haldar, Christian H. Stork, Michael Franz
~~September 2002~~ **Proceedings of the 2002 workshop on New security paradigms**
Full text available: pdf(507.59 KB) Additional Information: full citation, abstract, references, citings, index terms

We challenge the apparent consensus for using bytecode verification and techniques related to proof-carrying code for mobile code security. We propose an alternative to these two techniques that transports programs at a much higher level of abstraction. Our high-level encoding can achieve safe end-to-end transport of program source semantics. Moreover, our encoding is safe by construction, in the sense that unsafe programs cannot even be expressed in it. We contrast our encoding with certifying ...

Keywords: abstract syntax trees, language-based security, mobile code

15 Code Generation and Storage Allocation for Machines with Span-Dependent Instructions
Edward L. Robertson
January 1979 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 1 Issue 1
Full text available: pdf(852.48 KB) Additional Information: full citation, abstract, references, citings, index terms

Many machine languages have two instruction formats, one of which allows addressing of "nearby" operands with "short" (e.g. one word) instructions, while "faraway" operands require "long" format (e.g. two words). Because the distance between an instruction and its operand depends upon the formats of the intervening instructions, the formats of different instructions may be interdependent. An efficient technique is discussed which op ...

16 Automatic reasoning about numerical stability of rational expressions
B. W. Char
July 1989 **Pr ceedings f the ACM-SIGSAM 1989 international symp sium n Symb lic and algebraic c mputati n**
Full text available: pdf(688.11 KB) Additional Information: full citation, abstract, references, index terms, review

While numerical (e.g. Fortran) code generation from computer algebra is nowadays relatively easy to do, the expressions as they are produced via computer algebra typically benefit from non-trivial reformulation for efficiency and numerical stability. To assist in automatic "expert reformulation", we desire good automated tools to assess the stability of a particular form of an expression. In this paper, we discuss an approach to proofs of numerical stability (with respect to rou ...

17 Evolutionary design of complex software (EDCS) demonstration days 1999
Wayne Stidolph
January 2000 **ACM SIGSOFT Software Engineering Notes**, Volume 25 Issue 1
Full text available: pdf(1.90 MB) Additional Information: full citation, abstract, index terms

This report summarizes the Product/Technology demonstrations given at Defense Advanced Research Projects Agency (DARPA) Evolutionary Design of Complex Software (EDCS) Program Demonstration Days, held 28-29 June 1999 at the Sheraton National Hotel, Arlington, VA.

18 Parsing and compiling using Prolog
Jacques Cohen, Timothy J. Hickey
March 1987 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 9 Issue 2
Full text available: pdf(2.83 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper presents the material needed for exposing the reader to the advantages of using Prolog as a language for describing succinctly most of the algorithms needed in prototyping and implementing compilers or producing tools that facilitate this task. The available published material on the subject describes one particular approach in implementing compilers using Prolog. It consists of coupling actions to recursive descent parsers to produce syntax-trees which are subsequently utilized ...

19 A complete and practical algorithm for geometric theorem proving (extended abstract)
Ashutosh Rege
September 1995 **Proceedings of the eleventh annual symposium on Computational geometry**
Full text available: pdf(1.09 MB) Additional Information: full citation, references, citings, index terms

20 Specification, validation, and synthesis of email agent controllers: A case study in function rich reactive system design
Robert J. Hall
August 2000 **Proceedings of the third workshop on Formal methods in software practice**
Full text available: pdf(527.90 KB) Additional Information: full citation, abstract, references, index terms

With a few exceptions, previous formal methods for reactive system design have focused on finite state machines represented in terms of boolean states and boolean next-state functions. By contrast, in many reactive system domains requirements engineers and developers think in terms of complex data types and expressive next-state functions. Formal methods for reactive system design must be extended to meet their needs as well. I term a reactive system function rich if expr ...

Keywords: Electronic Mail, Formal Methods, Reactive Systems